2 - 3 , 4 - ,

[3]

object final (equals, hashCode, toString, clone, finalize) overriding .

(HashMap, HashSet) .

Object .

Comparable.compareTo Object .

item10 - equals

equals .(…)

Default .

So, ..!

- .
- ‘ (logical equality)’ .
- equals .
- private package-private equals .

equals

```
@Override public boolean equals(Object o){
    throw new AssertionError(); //!
}
```

?

-

ex) (Integer, String…) !

but, OK.

ex)Enum =

Object

equals .

- : null x x.equals(x) true.
- : null x, y x.equals(y) true y.equals(x) true.
- : null x, y, z x.equals(y) true y.equals(z) true x.equals(z) true.
- : null x, y x.equals(y) .
- null : null x x.equals(null) false.

: .

equals .

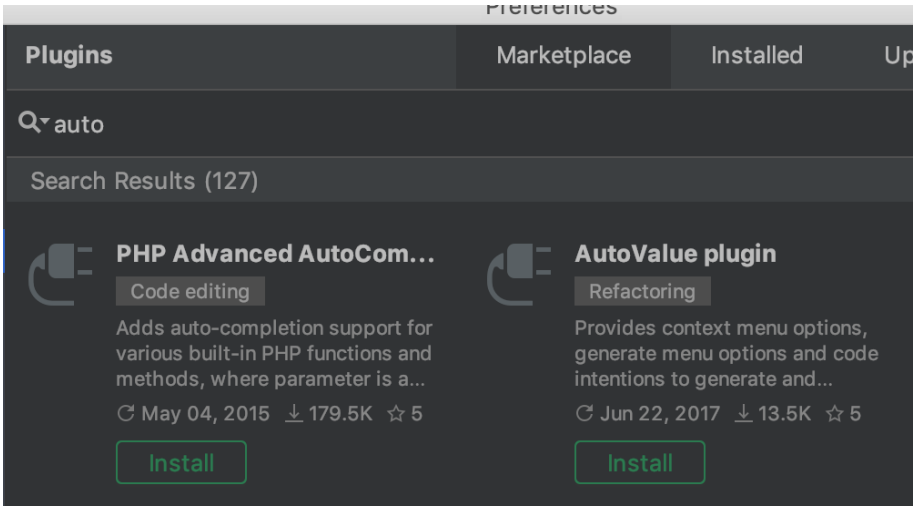
equals

1. == .
2. instanceof .
3. .
4. ‘’ .

, equals .

equals . ? ? ?

. (AutoValue)



- equals hashCode .
- .
- Object equals .

```
//   -    Object   !
//   Object.equals     .
//  equals     ,    equals   .
public boolean equals(MyClass o){
   ...
}
```

item11 - equals hashCode

equals **hashCode** .

hashCode . , **.**

Object

- hashCode .
- equals(Object) hashCode .
- hashCode . but, . .

hashCode

1. int result c .
2. f .

   a. c .

   b. c result . result = 31* result + c;
3. result .

.(AutoValue)

OK. equals .

.

hashCode .( )

.

hashCode API . .

item12 - toString

toString _@16__ .

.

toString .

why? toString , !

toString println, printf, (+), assert , . (!)

```
// PhoneNumber toString .
System.out.println(phoneNumber + "  .");
```

toString .

toString .

, .(ex BigInteger, BigDecimal)

/ . ___.

toString API .

- API .

…

(4) toString . (34) toString .

. (toString .)

AutoValue toString . Object toString .

item13 - clone

Clonable ?

Object protected clone .

clone ()

.

Clone . , clone .

..

Cloneable clone . -> '' , .

how? clone .

' ' .

(, clone ,)

item14 - Comparable

Comparable compareTo .

Object equals ?

compareTo , . .

```
Arrays.sort(a); // comparable
```

Comparable .

compareTo equals .

Comparable compareTo .(X)

compareTo .

compareTo -> 7 compare .

< > .

. .

Comparator . .

[4]

!! .

,, .

item15 -

?

. (,) -> **API**

- .
- .
- .
- .
- .

? !

.

.

.

- private
- package-private
- protected
- public

9

(export) public X..

. ( )

ex)JDK .

API , private.

API .

public public static final public .

public static final .

item16 - public public

```
//    public   !
//
class Point{
  public double x;
  public double y;
}
```

API , , ...

public , .

```
//      .
// public      !
class Point{
  private double x;
  private double y;

  public Point(double x, double y){
    this.x = x;
    this.y = y;
  }

  public double getX() { return x; }
  public double getY() { return y; }

  public void setX( double x ) { this.x = x; }
  public void setX( double y ) { this.y = y; }
}
```

.

package-private private . .

why? .

⚠️ : java.awt.package Point Dimension. X ..

item17 -

: . .

ex) String, , BigInteger, BigDecimal .

, .

- () .
- .
- final .
- private .
- .

: private package-private public

```
public final class Complex {
    private final double re;
    private final double im;

    public static final Complex ZERO = new Complex(0, 0);
    public static final Complex ONE  = new Complex(1, 0);
    public static final Complex I    = new Complex(0, 1);

    public Complex(double re, double im) {
        this.re = re;
        this.im = im;
    }

    public double realPart()      { return re; }
    public double imaginaryPart() { return im; }

    //
    public Complex plus(Complex c) {  //  add   plus
        //        Complecx
        return new Complex(re + c.re, im + c.im);
    }

    //  17-2  (private    .) (110-111)
    public static Complex valueOf(double re, double im) {
        return new Complex(re, im);
    }

    public Complex minus(Complex c) {
        return new Complex(re - c.re, im - c.im);
    }

    public Complex times(Complex c) {
        return new Complex(re * c.re - im * c.im,
                re * c.im + im * c.re);
    }

    public Complex dividedBy(Complex c) {
        double tmp = c.re * c.re + c.im * c.im;
        return new Complex((re * c.re + im * c.im) / tmp,
                (im * c.re - re * c.im) / tmp);
    }

    @Override public boolean equals(Object o) {
        if (o == this)
            return true;
        if (!(o instanceof Complex))
            return false;
        Complex c = (Complex) o;

        // ==  compare   63 .
        return Double.compare(c.re, re) == 0
                && Double.compare(c.im, im) == 0;
    }
    @Override public int hashCode() {
        return 31 * Double.hashCode(re) + Double.hashCode(im);
    }

    @Override public String toString() {
        return "(" + re + " + " + im + "i)";
    }
}
```

=

= ,

, .

.

.

.(!) So, :)

(1) , .

(50) .

. .

?

. ....

- getter setter . .
- .
- private final .
- , .

public .

. -> .

Item 18 :

, . .

- .

.

```
public class InstrumentedHashSet<E> extends HashSet<E> {
    //
    private int addCount = 0;

    public InstrumentedHashSet() {
    }

    public InstrumentedHashSet(int initCap, float loadFactor) {
        super(initCap, loadFactor);
    }

    @Override public boolean add(E e) {
        addCount++;
        return super.add(e);
    }

    @Override public boolean addAll(Collection<? extends E> c) {
        addCount += c.size();
        return super.addAll(c);
    }

    public int getAddCount() {
        return addCount;
    }

    public static void main(String[] args) {
        InstrumentedHashSet<String> s = new InstrumentedHashSet<>();
        s.addAll(List.of("Snap", "Crackle", "Pop"));
        System.out.println(s.getAddCount());
    }

}
```

: 3

: 6

InstrumentedHashSet addAll() HashSet addAll() .

addAll method in HashSet

```
public boolean addAll(Collection<? extends E> c) {
        boolean modified = false;
        for (E e : c)
            if (add(e))
                modified = true;
        return modified;
    }
```

add() .

- ***?***

  1. addAll HashSet addAll add .

  2. addAll HashSet addAll ( )

  3. (, return type)\

  4. , private (composition .)

```
public class InstrumentedSet<E> extends ForwardingSet<E> {
    private int addCount = 0;

    public InstrumentedSet(Set<E> s) {
        super(s);
    }

    @Override public boolean add(E e) {
        addCount++;
        return super.add(e);
    }
    @Override public boolean addAll(Collection<? extends E> c) {
        addCount += c.size();
        return super.addAll(c);
    }
    public int getAddCount() {
        return addCount;
    }

    public static void main(String[] args) {
        InstrumentedSet<String> s = new InstrumentedSet<>(new HashSet<>());
        s.addAll(List.of("Snap", "Crackle", "Pop"));
        System.out.println(s.getAddCount());
    }
}


public class ForwardingSet<E> implements Set<E> {
    private final Set<E> s;
    public ForwardingSet(Set<E> s) { this.s = s; }

    public void clear()               { s.clear();            }
    public Iterator<E> iterator()     { return s.iterator();  }
    public boolean add(E e)           { return s.add(e);      }
    public boolean remove(Object o)   { return s.remove(o);   }
    public boolean addAll(Collection<? extends E> c)
    { return s.addAll(c);      }
        //...


}
```

- Set InstrumentedSet **wrapper** Set , **Decorator pattern** .

- **is-a** . (ex. A B : B A?)
- Java **Stack Vector**, **Properties HashTable** .

Item 19 -

- .

```
/**
* {@inheritDoc}
*
* @implSpec
* This implementation iterates over the collection looking for the
* specified element. If it finds the element, it removes the element
* from the collection using the iterator's remove method.
*
* <p>Note that this implementation throws an
* {@code UnsupportedOperationException} if the iterator returned by this
* collection's iterator method does not implement the {@code remove}
* method and this collection contains the specified object.
*
* @throws UnsupportedOperationException {@inheritDoc}
* @throws ClassCastException {@inheritDoc}
* @throws NullPointerException {@inheritDoc}
*/
```

iterator remove

- .
  - .

```
// Class whose constructor invokes an overridable method. NEVER DO THIS! (Page 95)
public class Super {
    // Broken - constructor invokes an overridable method
    public Super() {
        overrideMe();
    }

    public void overrideMe() {
    }
}
```

```
// Demonstration of what can go wrong when you override a method  called from constructor (Page 96)
public final class Sub extends Super {
    // Blank final, set by constructor
    private final Instant instant;

    Sub() {
        instant = Instant.now();
    }

    // Overriding method invoked by superclass constructor
    @Override public void overrideMe() {
        System.out.println(instant);
    }

    public static void main(String[] args) {
        Sub sub = new Sub();
        sub.overrideMe();
    }
}
```

- .
- final , .

item 20 :

-
- **(mixin) .**
  - mixin : , primary type (?)

    ex) Comparable mixin interface
- .
  - (ex singer, songwriter)

```
public interface Singer {
    AudioClip sing(Song s);
}


public interface Songwriter {
    Song compose(boolean hit);
}


public interface SingerSongwriter extends Songwriter, Singer{
   AudioClip strum();
   void actSensitive();
}
```

- (skeletal implementation) 2 .
  - ?

-

Item 21 :

- 8 .
  -
- 8 < >.

ex) apache commons SynchronizedCollection

- .

Item 22 :

-
-

ex) (- public static final)

```
// Constant interface antipattern - do not use!
public interface PhysicalConstants {
    // Avogadro's number (1/mol)
    static final double AVOGADROS_NUMBER   = 6.022_140_857e23;

    // Boltzmann constant (J/K)
    static final double BOLTZMANN_CONSTANT = 1.380_648_52e-23;

    // Mass of the electron (kg)
    static final double ELECTRON_MASS      = 9.109_383_56e-31;
}
```

. .

Integer.MAX_VALUE, **enum type**, . .

```
// Constant utility class (Page 108)
public class PhysicalConstants {
  private PhysicalConstants() { }  // Prevents instantiation

  // Avogadro's number (1/mol)
  public static final double AVOGADROS_NUMBER = 6.022_140_857e23;

  // Boltzmann constant (J/K)
  public static final double BOLTZMANN_CONST  = 1.380_648_52e-23;

  // Mass of the electron (kg)
  public static final double ELECTRON_MASS    = 9.109_383_56e-31;
}
```

Item 23 :

- .

ex) tagged class

tagged class

```
class Figure {
    enum Shape { RECTANGLE, CIRCLE };

    // Tag field - the shape of this figure
    final Shape shape;

    // These fields are used only if shape is RECTANGLE
    double length;
    double width;

    // This field is used only if shape is CIRCLE
    double radius;

    // Constructor for circle
    Figure(double radius) {
        shape = Shape.CIRCLE;
        this.radius = radius;
    }

    // Constructor for rectangle
    Figure(double length, double width) {
        shape = Shape.RECTANGLE;
        this.length = length;
        this.width = width;
    }

    double area() {
        switch(shape) {
            case RECTANGLE:
                return length * width;
            case CIRCLE:
                return Math.PI * (radius * radius);
            default:
                throw new AssertionError(shape);
        }
    }
}
```

- : .,, .

```
abstract class Figure {
    abstract double area();
}

class Circle extends Figure {
    final double radius;

    Circle(double radius) { this.radius = radius; }

    double area() { return Math.PI * (radius * radius); }
}

class Rectangle extends Figure {
    final double length;
    final double width;

    Rectangle(double length, double width) {
        this.length = length;
        this.width = width;
    }

    double area() { return length * width; }
}
```

- , .
-

Item 24 : static .

- nested class . nested class .
- , , , .
- static . .
- GC .

Item 25 :

- .
- ___ ___ .

ex)

main.java

```
public class Main {
    public static void main(String[] args) {
        System.out.println(Utensil.NAME + Dessert.NAME);
    }
}
```

Utensil.java

```
class Utensil {
    static final String NAME = "pan";
}

class Dessert {
    static final String NAME = "cake";
}
```

Dessert.java

```
// Two classes defined in one file. Don't ever do this! (Page 115)
class Utensil {
    static final String NAME = "pot";
}

class Dessert {
    static final String NAME = "pie";
}
```

- **javac Main.java Dessert.java ： ?**
- **javac Main.java or javac Main.java Utensil.java ： ?**
- **javac Dessert.java Main.java： ?**